

5. LIMBAJUL SQL

5.1 Prezentare generală

SQL (Structured Query Language) este în prezent, unul din cele mai puternice limbaje structurate pentru interogarea bazelor de date relaționale.

Pronunția oficială: „si-q-el”. Neoficial, „si-quel”.

Este un limbaj **neprocedural** și **declarativ**, deoarece utilizatorul descrie *ce* date vrea să obțină, fără a fi nevoie să stabilească modalitățile de a ajunge la datele respective. Nu poate fi considerat un limbaj de programare sau unul de sistem, ci mai degrabă face parte din categoria *limbajelor de aplicații*, fiind *orientat pe mulțimi*. Foarte frecvent, este utilizat în administrarea bazelor de date client/server, aplicația client fiind cea care generează instrucțiunile SQL.

Lansat inițial de IBM. Standardizat prima dată de ANSI, apoi ISO. Actualmente, ISO 92.

Pentru că există o standardizare a limbajului SQL, multe SGBD (Oracle, Access, Sybase) recunosc principalele instrucțiuni ale acestuia.

Caracteristicile adăugate standardului se numesc **extensii**. De ex, în standard sunt specificate 6 tipuri diferite de date pentru o BD SQL. În multe implementări, această listă este completată cu o diversitate de extensii. Fiecare implementare se numește **dialect**. Dialectul ACCSES conține unele particularități, fiind conceput mai mult pentru crearea interogărilor de selecție.

Există 3 metode de bază privind **implementarea limbajului SQL**:

- **apelare directă** (Direct Invocation): constă în introducerea instrucțiunilor direct de la prompter
- **modulară** (Modul Language): folosește proceduri apelate de programele aplicație
- **încapsulată** (Embedded SQL): conține instrucțiuni încapsulate în codul de program

Instrucțiunile SQL pot fi grupate în:

- instrucțiuni de **definire** a datelor, care permit descrierea structurii BD
- instrucțiuni de **manipulate** a datelor: adaugă, șterge, modifică înregistrări
- instrucțiuni de **selecție** a datelor, care permit consultarea BD
- instrucțiuni de **procesare** a tranzacțiilor
- instrucțiuni de **control al cursorului**
- instrucțiuni privind **controlul accesului la date**

În limbajul SQL standardizat de ISO nu se folosesc termenii formali de *relație*, *atribut*, *tuplu*, ci **tabel**, **coloană**, **rând**.

Scrierea comenzilor SQL

O instrucțiune SQL este formată din **cuvinte rezervate** și **cuvinte definite de utilizator**. Cuvintele rezervate constituie partea fixă și se scriu *exact* cum este necesar. Cuvintele definite de utilizator reprezintă denumirile diverselor obiecte din BD.

Deși standardul nu o cere, majoritatea dialectelor cer **terminator de instrucțiune** („;”).

Majoritatea componentelor nu sunt sensibile la tipul de litere (excepție importantă: când datele au caracter literal, de ex, dacă se stochează numele „POPA” și căutăm „Popa”, nu vom găsi înregistrarea respectivă).

Deși SQL este un limbaj cu **format liber**, o instrucțiune este mai **lizibilă** dacă se utilizează **indentarea** și **aliniera**. De ex:

- **fiecare clauză** dintr-o instrucțiune trebuie să înceapă pe o linie nouă
- **începutul fiecărei clauze** să fie aliniat cu începutul celorlalte
- dacă o **clauză are mai multe părți**, **fiecare parte** trebuie să apară pe câte o linie separată și trebuie indentată față de începutul clauzei

Convenții de notare folosite în definirea instrucțiunilor:

- **majuscule** pentru cuvintele rezervate
- **litere mici** pentru cuvinte definite de utilizator
- **bara verticală** | indică posibilitatea alegerii dintre mai multe variante
- **acoladele** { } indică un element necesar
- **parantezele drepte** [] indică un element opțional
- **punctele de suspensie** ... indică o repetare opțională a unui articol, de 0 sau mai multe ori

Identificatorii SQL sunt utilizați pentru a numi obiecte din BD. Pentru caracterele utilizate, standardul ISO permite A...Z, a...z, 0...9, **_**. **Restricții** impuse identificatorilor:

- nu poate fi mai lung de 128 caractere (majoritatea dialectelor au o limită mult mai joasă)
- trebuie să înceapă cu o literă
- nu poate conține spații libere

5.2 Instrucțiuni pentru definirea datelor

Teoretic, comenzile pentru definirea datelor fac parte din modulul corespunzător componentei DDL din SGBD. Totuși, în majoritatea implementărilor SQL comenzile de definire a datelor sunt prelucrate de același interpretor care rezolvă și interogările și operațiile de manipulare a datelor. Componentele DML și DDL ale SGBD sunt implementate în același modul software.

CREATE DATABASE nume_bd

Crează o bază de date. Majoritatea SGBD permit crearea unei BD print-un simplu *clic* de mouse. Există și posibilitatea folosirii acestei instrucțiuni, dar mult mai greoi. Comanda nu e standardizată, ACCESS nici nu o acceptă.

CREATE TABLE nume_tabel

(câmp1 tip_dată [NOT NULL],
câmp2 tip_dată [NOT NULL],
câmp3 tip_dată [NOT NULL]...)

Crează un tabel și definește structura unei înregistrări precum și tipurile de date asociate câmpurilor. Numele tabelului trebuie să fie unic în BD. Clauza NOT NULL arată că în câmpul respectiv nu se memorează valori de tip NULL.

ALTER TABLE nume_tabel

ADD nume_câmp tip_dată

Adaugă un câmp la un tabel existent. Ștergerea unui câmp nu este posibilă.

DROP TABLE nume_tabel

Șterge complet un tabel din BD.

DROP DATABASE nume_bd

Șterge BD. Există însă o multitudine de restricții stabilite de administratorul sistemului privind această operație. Multe versiuni SQL nu includ această instrucțiune, ștergerea făcându-se din comenzi de mouse.

5.3 Instrucțiuni pentru selecția datelor

5.3.1 Cereri de interogare simple

Instrucțiunile de selecție reprezintă una din categoriile cele mai importante ale limbajului SQL ACCESS. Indiferent dacă sunt cereri simple sau complexe, cuvântul cheie este **SELECT**. Pentru cererile de interogare simple, sintaxa instrucțiunii este:

```
SELECT [domeniu] listă_selecție  
      FROM nume_tabel1, nume_tabel2,...  
      [WHERE criteriu_selecție]  
      [ORDER BY câmpuri_criteriu [ASC|DESC]];
```

□ Domeniu

Specifică o opțiune de **includere** sau **eliminare** din rezultatul selecției, a înregistrărilor care conțin duplicate. Opțiunile posibile sunt:

ALL cere includerea tuturor înregistrărilor care îndeplinesc condițiile impuse. Cum instrucțiunile *SELECT tabel* și *SELECT ALL tabel* au același rezultat practic, calificativul **ALL** este rar folosit.

DISTINCT cere eliminarea înregistrărilor care conțin duplicate în câmpurile selectate, afișând numai o apariție a acesteia.

DISTINCTROW cere eliminarea înregistrărilor care conțin duplicate în ansamblul lor, nu numai în câmpurile selectate, afișând numai o apariție a acesteia.

- **Listă_selecție** cuprinde câmpurile care dorim să apară în tabelul cu rezultatele interogării. Similar cu *Field ... Show ...* din grila de proiectare QBE.
- **Clauza FROM** specifică numele **tabelului** sau tabelelor pe care se face cererea de interogare. Pentru mai multe tabele, numele acestora se separă cu „,”. Pe lângă tabele, ca sursă de informații pot apare și **interogări** deja create.
- **Clauza WHERE** cere numai înregistrările care îndeplinesc criteriul de selecție specificat. Criteriul de selecție este o expresie care conține obligatoriu și un **operator adecvat** tipului de dată al câmpului respectiv. Clauza **WHERE** este opțională.
- **Clauza ORDER BY** cere ordonarea în mod crescător (**ASC**) sau descrescător (**DESC**) a rezultatelor interogării. Ordonarea este opțională și se poate face după unul sau mai multe câmpuri_criteriu.

5.3.2 Cereri de interogare complexe

Sunt acele interogări în care apar funcțiile **agregat**, **asocierile** sau **combinările**.

Funcțiile agregat (de grup)

Permit construirea unor interogări complexe, prin care utilizatorul cere gruparea înregistrărilor care au câmpuri cu aceeași valoare, în scopul efectuării unor calcule. În standardul ISO sunt definite 5 funcții de grup:

COUNT returnează numărul de valori dintr-o coloană specificată
SUM returnează suma valorilor dintr-o coloană specificată
AVG returnează media valorilor dintr-o coloană specificată
MIN returnează cea mai mică valoare dintr-o coloană specificată

MAX returnează cea mai mare valoare dintr-o coloană specificată

Sintaxa instrucțiunii:

```
SELECT [domeniu] funcție_agregat(nume_câmp) AS alias [,listă_selecție]
      FROM nume_tabel1, nume_tabel2,...
      GROUP BY câmp_de_grupare
      [HAVING criteriu_de_grupare]
      [ORDER BY câmpuri_criteriu [ASC|DESC]];
```

Elementele noi de sintaxă:

- ❑ **AS alias** asociază un pseudonim rezultatului funcției agregat
- ❑ **Clauza GROUP BY** precizează **câmpul** sau **câmpurile** după care se face gruparea înregistrărilor. Echivalentul acestei clauze în macheta grafică QBE îl reprezintă rândul *Total*.
- ❑ **Clauza HAVING** conține criteriul care va fi aplicat **câmpului argument** al funcției agregat. Spre deosebire de *WHERE*, care *acționează înainte de gruparea* înregistrărilor, **HAVING acționează după definirea** acesteia.

Asocierile (interogările JOIN)

Limbajul SQL oferă posibilitatea de a grupa și folosi date din tabele diferite. Operațiile de asociere induse de **clauza JOIN** au ca rezultat producerea tuturor combinațiilor posibile, pentru conținutul informațional al fiecărui tabel. Noile înregistrări care rezultă în urma joncțiunii sunt disponibile pentru selecțiile următoare. La o asociere pot participa mai mult de 2 tabele.

Există mai multe categorii de joncțiuni:

- **CROSS** (încrucișată) – rar folosită
- **ECHIVALENTĂ** (echijoncțiune) – cea mai folosită – presupune folosirea **clauzei WHERE** asociată cu o egalitate dorită
- **NEECHIVALENTĂ** (non echijoncțiune) - rar folosită - presupune folosirea **clauzei WHERE** asociată cu orice alt operator de comparare, în afară de „=”

Sintaxa instrucțiunii pentru joncțiunile echivalente și neechivalente este:

```
SELECT [domeniu] listă_selecție
      FROM nume_tabel1, nume_tabel2,...
      [WHERE criteriu_de_asociere]
      [ORDER BY câmpuri_criteriu [ASC|DESC]];
```

Deoarece în instrucțiunile SQL care descriu joncțiuni se utilizează câmpuri care fac parte din tabele diferite, trebuie specificat numele tabelului de care aparțin, folosind sintaxa

nume_tabel.nume_câmp

fără spații înainte sau după punct.

După modul de asociere a înregistrărilor din tabele, joncțiunile pot fi:

- **interne** sau **INNER JOIN** determină o asociere a înregistrărilor din tabele, astfel încât să rezulte un număr total de înregistrări egal cu **produsul** numărului de înregistrări din fiecare tabel
- **externe de stânga** sau **LEFT OUTER JOIN**
- **externe de dreapta** sau **RIGHT OUTER JOIN**

Echivalentul QBE al acestor categorii de joncțiuni este alegerea opțiunilor 1, 2, sau 3 din caseta Join Properties.

Sintaxa instrucțiunii:

```
SELECT [domeniu] listă_selecție
```

```

FROM nume_tabel1
    {INNER|LEFT OUTER|RIGHT OUTER} JOIN nume_tabel2
        ON criteriu_de_asociere
    [{INNER|LEFT OUTER|RIGHT OUTER} JOIN nume_tabel3
        ON criteriu_de_asociere]...
    [WHERE criteriu_selecție]
    [ORDER BY câmpuri_criteriu [ASC|DESC]];

```

Obs: SQL ACCESS acceptă scrierea fără specificarea explicită a lui OUTER.

Semnificația elementelor noi:

- ❑ **JOIN** specifică tabelul care va fi asociat (nume_tabel2, nume_tabel3) celui din clauza FROM
- ❑ **ON** arată între ce câmpuri trebuie să existe relația pe care se bazează joncțiunea. Criteriul de asociere conține obligatoriu operatorul „=”.

Combinările (interogările UNION)

Când utilizatorul dorește să vadă rezultatele mai multor interogări SELECT în același timp, prin combinarea ieșirilor lor, se poate utiliza facilitatea UNION. De remarcat că nu există echivalent QBE pentru această instrucțiune.

Sintaxa generală pentru interogările UNION este:

```

SELECT listă_câmpuri FROM nume_tabel1
    UNION SELECT listă_câmpuri FROM nume_tabel2
        [GROUP BY câmp_de_grupare]
        [HAVING criteriu_de_agregare]
    [UNION SELECT listă_câmpuri FROM nume_tabel3
        [GROUP BY câmp_de_grupare]
        [HAVING criteriu_de_agregare]]
    [UNION ...]
    [ORDER BY câmpuri_criteriu [ASC|DESC]];

```

Există mai multe restricții pentru instrucțiunile care generează interogările UNION, și anume:

- Numărul câmpurilor din lista de câmpuri din fiecare instrucțiune SELECT și UNION SELECT trebuie să fie aceeași
- Secvența de nume din fiecare listă de câmpuri trebuie să corespundă unor intrări identice
- Este permisă utilizarea doar o dată a clauzei ORDER BY, după ultima instrucțiune UNION SELECT

Exemple:

```

SELECT nume, prenume, vârstă FROM Colaboratori2001
    UNION SELECT nume, prenume, vârstă FROM Colaboratori2002
    ORDER BY nume;

```

```

SELECT nume, prenume, vârstă FROM Colaboratori2001
    GROUP BY categoria
    HAVING categoria="student"
    UNION SELECT nume, prenume, vârstă FROM Colaboratori2002
    GROUP BY categoria
    HAVING categoria="student"

```

5.4 Instrucțiuni pentru manipularea datelor

Foarte utile în exploatarea unei BD, aceste instrucțiuni se implementează prin interogările **de acțiune**. Este necesară o mare precauție în utilizarea lor deoarece acțiunile sunt **ireversibile**, putând influența inclusiv integritatea referențială a BD.

Cele mai importante sunt: CREATE, INSERT, UPDATE și DELETE.

- Interogările de acțiune tip **CREATE** duc la generarea unui nou tabel pornind de la structura și conținutul unor tabele deja existente. Se folosește instrucțiunea **SELECT ... INTO**

```
SELECT [domeniu] (câmp1,câmp2...)  
    INTO tabel_nou  
    FROM tabel_sursa  
    [WHERE criteriu_de_adăugare];
```

- Interogările de acțiune tip **INSERT** sunt folosite pentru adăugarea de înregistrări dintr-un tabel în altul. Există două forme ale instrucțiunii și anume:

- **INSERT ... VALUES**
- **INSERT ... SELECT**

a). În primul caz se adaugă *o singură înregistrare într-un tabel*, menționându-se câmpurile și valorile acestora. Se utilizează pentru operații simple, care presupun lucrul cu un număr redus de înregistrări.

```
INSERT INTO nume_tabel (câmp1, câmp2...)  
    VALUES (valoare1, valoare2...);
```

Reguli:

- valorile din clauza VALUES vor avea aceeași natură cu câmpurile din clauza INTO
- mărimea valorii va fi < dimensiunea câmpului
- corespondență între câmp1 și valoare1, etc.
- Dacă un câmp are specificația NOT NULL, este obligatorie introducerea unei valori pentru aceasta

b). În al doilea caz, este posibil să se copieze *mai multe înregistrări* dintr-un tabel în unul sau mai multe tabele.

```
INSERT INTO tabel_destinație (câmp1, câmp2...)  
    SELECT [domeniu] câmp1, câmp2...  
    FROM tabel_sursă  
    WHERE criteriu_de_adăugare;
```

Reguli:

- aceleași ca mai sus
- numărul și natura câmpurilor din clauza **INTO** să fie aceleași cu cele returnate de instrucțiunea **SELECT**
- dacă nu se introduce **WHERE**, toate înregistrările din **tabel_sursă** vor fi adăugate în **tabel_destinație**

- Interogările de acțiune tip **DELETE** șterg parțial sau total înregistrările dintr-un tabel. Nu se folosește pentru ștergerea de valori din câmpuri individuale, ci acționează asupra înregistrării în totalitatea ei. Dacă se șterg toate înregistrările, structura de tabel rămâne, ea putând fi eliminată numai cu DROP TABLE.

**DELETE FROM nume_tabel
[WHERE criteriu_de_ștergere];**

Ca și instrucțiunea INSERT, operația de ștergere a înregistrărilor dintr-o tabelă poate duce la probleme de integritate referențială în alte tabele.

Exemplu:

**DELETE *
FROM Vânzări**

**DELETE *
FROM Angajați
WHERE Vârsta>60**

- Interogările de acțiune tip UPDATE pot introduce înregistrări noi și pot modifica valorile câmpurilor din înregistrări existente.

**UPDATE nume_tabel
SET nume_câmp1=valoare1 [,nume_câmp2=valoare2]...
[WHERE criteriu_de_actualizare];**

Ca și în celelalte locuri unde apare clauza WHERE, restricționarea se poate accentua folosind și operatori logici.

Exemplu:

**UPDATE Comunicații
SET Rețea="Orange"
WHERE Rețea="Dialog" AND Data>#12.12.2001;**

5.5 Cereri de interogare imbricate

Scrierea unei interogări în cadrul alteia duce la apariția unei subinterogări, setul de rezultate obținute de la aceasta constituind argument pentru prima interogare.

**SELECT lista_câmpuri
FROM tabel1
WHERE tabel1.nume_câmp=
(SELECT nume_câmp
FROM tabel2
WHERE criteriu_de_selecție);**

Cele două tabele trebuie să aibă un câmp comun (nume_câmp) care va reprezenta câmpul de legătură ce stă la baza construirii subinterogării.